CODE PATH *ORG

Decision Structures and Boolean Logic

CIIC 3015

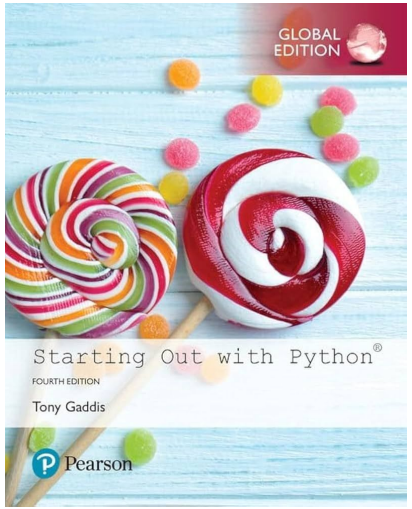
Karelys López Rivera

Lecture Goals

- **Relational Operators**
  - What are they?
  - Live Coding

- **Logical Operators**
  - What are they?
  - Examples
  - Live Coding

- **Conditionals**
  - What are they?
  - How do we write them?
  - Live Coding



Starting Out With Python (4th Edition) Chapter 3

Relational operators

Relational operators in Python

Relational operators compare values and return a Boolean result of `True` or `False` based on the comparison.

Meaning	Operator	Operator in Python
less than	<	<
greater than	>	>
equal to	=	==
less than or equal to	≤	<=
greater than or equal to	≥	>=
not equal to	≠	!=

Example: Relational operators

The == operator

```
apples = 10
oranges = 8
print(apples == oranges)
```

```
False
```

The < operator

```
apples = 10
oranges = 8
print(apples < oranges)
```

```
False
```

The <= operator

```
apples = 10
oranges = 8
print(apples <= oranges)
```

```
False
```

The != operator

```
apples = 10
oranges = 8
print(apples != oranges)
```

```
True
```

The > operator

```
apples = 10
oranges = 8
print(apples > oranges)
```

```
True
```

The >= operator

```
apples = 10
oranges = 8
print(apples >= oranges)
```

```
True
```

Check your understanding

What is the output of this code block?

```
shipment1 = 450
shipment2 = 500

quantity_check = shipment1 >= shipment2
print(quantity_check)
```

Check your understanding

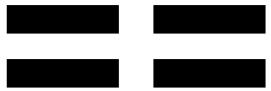
What is the output of this code block?

```
shipment1 = 450
shipment2 = 500

quantity_check = shipment1 >= shipment2
print(quantity_check)
```

```
False
```

Comparison Operators: Equals sign(s)!



Don’t confuse “=” with “==”!

assignment operator

think *gets the value*

my_var = 20

equality operator

think *is equivalent to?*

22 == 23

IT'S CODING TIME

- Less than <
- Greater than >
- Less than or equal to <=
- Greater than or equal to >=
- Equal to ==
- Not equal to !=

Questions?

Post to our Discord server

or

email:

upr-ciic3015-staff@upr.edu

karelys.lopez1@upr.edu

Give us anonymous feedback!

https://bit.ly/CIIC3015_F25_Feedback

CODE PATH *ORG

Logical operators

Logical operators

- **Logical operators** combine or modify Boolean values and return a new Boolean result.
    - Allow you to combine multiple conditions into a single expression that evaluates to `True` or `False`
- Python provides three fundamental logical operators:
    - `and`
    - `or`
    - `not`

The and operator

The `and` operator returns `True` only if both conditions are `True`.

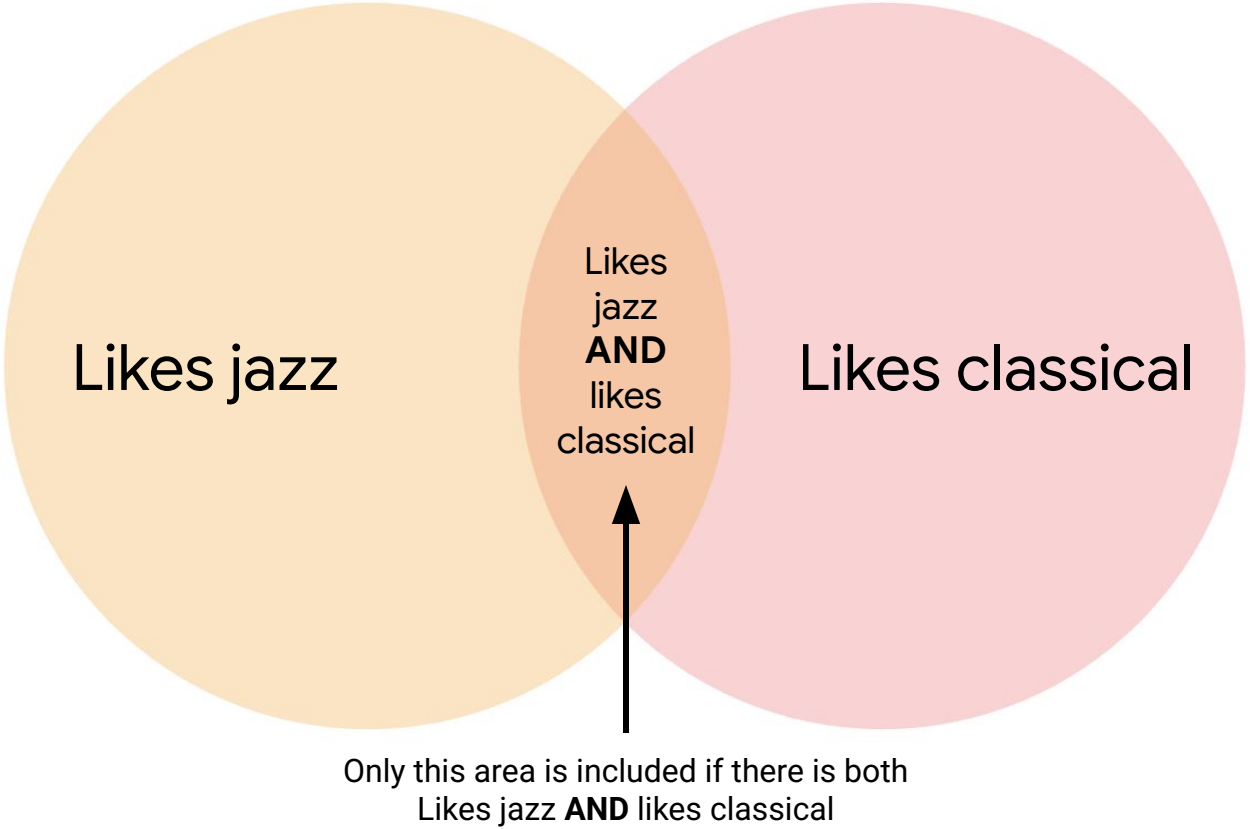
Condition 1	Condition 2	Result
`True`	`True`	`True`
`True`	`False`	`False`
`False`	`True`	`False`
`False`	`False`	`False`

Example: The and operator

The and operator returns `True` only if both conditions are `True`.

Likes jazz	Likes classical	Likes jazz **AND** likes classical
`True`	`True`	`True`
`True`	`False`	`False`
`False`	`True`	`False`
`False`	`False`	`False`

Example: The and operator



Examine the code

What is the output of this code block?

```
is_of_age = True
has_drivers_license = False
can_drive = is_of_age and has_drivers_license
print(can_drive)
```

Examine the code

What is the output of this code block?

```
is_of_age = True
has_drivers_license = False
can_drive = is_of_age and has_drivers_license
print(can_drive)
```

```
False
```

The `or` operator

The `or` operator returns `True` if at least one condition is `True`.

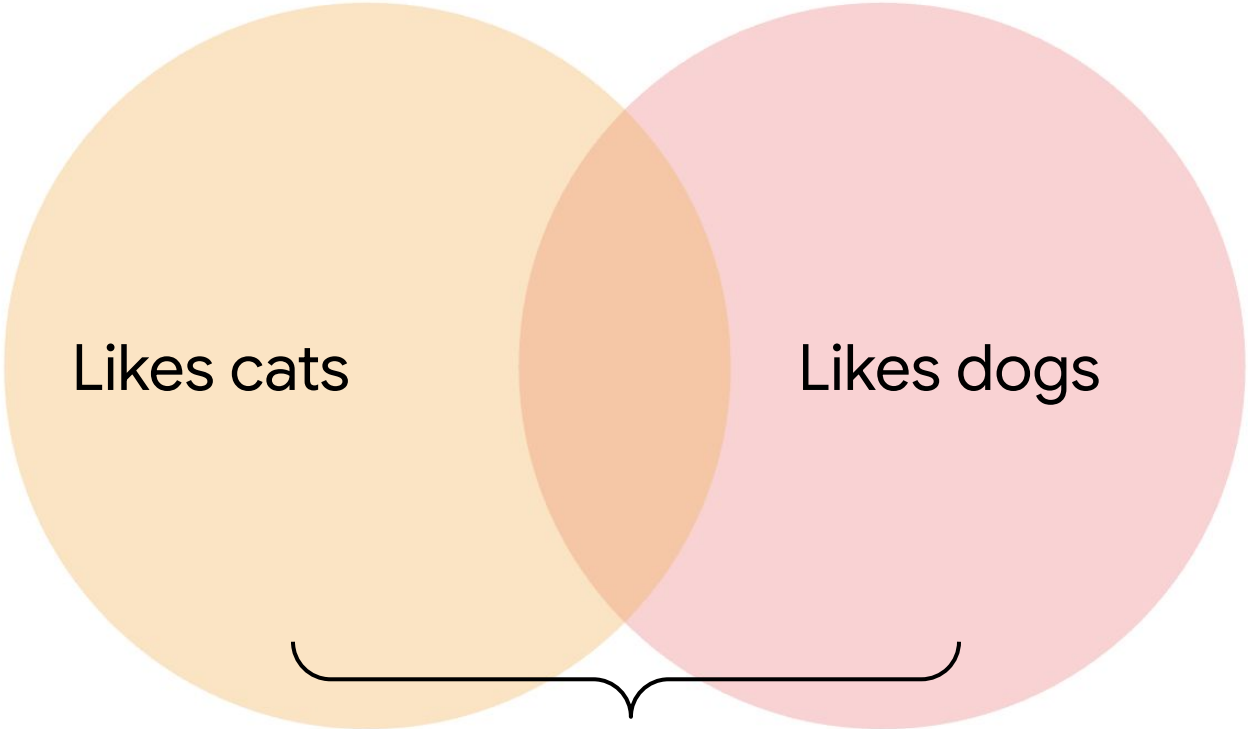
Condition 1	Condition 2	Result
`True`	`True`	`True`
`True`	`False`	`True`
`False`	`True`	`True`
`False`	`False`	`False`

Example: The `or` operator

The `or` operator returns `True` if at least one condition is `True`.

Likes cats	Likes dogs	Likes cats OR likes dogs
`True`	`True`	`True`
`True`	`False`	`True`
`False`	`True`	`True`
`False`	`False`	`False`

Example: The `or` operator



All three sections are included if there are either cats **OR** dogs

Examine the code

What is the output of this code block?

```
hiking = True
reading = False
is_exercise = hiking or reading
print(is_exercise)
```

Examine the code

What is the output of this code block?

```
hiking = True
reading = False
is_exercise = hiking or reading
print(is_exercise)
```

```
True
```

The `not` operator

The `not` operator reverses the value of a condition.

Condition	**Result**
`True`	`False`
`False`	`True`

Example: The `not` operator

The `not` operator reverses the value of a condition.

Is a sports fan	**Is NOT a sports fan**
`True`	`False`
`False`	`True`

Examine the code

What is the output of this code block?

```
blue = True
red = False
not_blue = not blue
not_red = not red
print(not_blue)
print(not_red)
```

Examine the code

What is the output of this code block?

```
blue = True
red = False
not_blue = not blue
not_red = not red
print(not_blue)
print(not_red)
```

```
False
True
```

Precedence

Precedence refers to the order in which logical operators are evaluated:

1. `not`
2. `and`
3. `or`

You can use parentheses to control the order. Logical operations within parentheses are evaluated first.

```
result = not True and False or True
print(result)
```

```
True
```

```
result = (not True) and (False or True)
print(result)
```

```
False
```

Combining Logical Operators

Remember PEMDAS? Order of operations for arithmetic operators

Similarly...

Parentheses

not

and

or

By: Jean Carlos Méndez

Practice: Combining Logical Operators

(Parentheses, not, and, or)

```
True and not False
```

```
True and True
```

True

Protip: watch the animations of this slide using slideshow

By: Jean Carlos Méndez

Google | GR |

Practice: Combining Logical Operators

(Parentheses, not, and, or)

```
(False or True) and False
```

```
True and False
```

False

Protip: watch the animations of this slide using slideshow

By: Jean Carlos Méndez

Practice: Combining Logical Operators

(Parentheses, not, and, or)

```
not (True or False) or True
    not True or True
      False or True
           True
```

Protip: watch the animations of this slide using slideshow

By: Jean Carlos Méndez

Google | GLR |

Example: Logical operators

```
day1 = "Sunny"
day2 = "Cloudy"
day3 = "Stormy"

print(not day1 == "Sunny" and day2 == "Cloudy" or day3 == "Stormy")
print(not(day1 == "Sunny" and day2 == "Cloudy" or day3 == "Stormy"))

True
False
```

Check your understanding

What is the output of this code block?

```
product1 = "laptop"
product2 = "smartphone"
product3 = "tablet"

print(not (not product1 == "laptop" or product2 == "smartphone") and product3 == "tablet")
```

Check your understanding

What is the output of this code block?

```
product1 = "laptop"
product2 = "smartphone"
product3 = "tablet"

print(not (not product1 == "laptop" or product2 == "smartphone") and product3 == "tablet")
```

```
False
```

IT'S CODING TIME

- and
- or
- not
- or & and
- and & not
- or & not
- and, or & not

Questions?

Post to our Discord server

or

email:

upr-ciic3015-staff@upr.edu

karelys.lopez1@upr.edu

Give us anonymous feedback!

https://bit.ly/CIIC3015_F25_Feedback

CODE PATH *ORG

Conditional statements

Conditional statements

A **conditional statement** evaluates code to determine whether it meets a specific condition (or set of conditions).

- When a condition is met, it evaluates to a Boolean value of `True` and performs specified actions
- When the condition isn't met, it evaluates a Boolean value of `False` and doesn't perform the specified actions

Common types of conditional statements in Python:

- `if` statement
- `if-else` statement
- `if-elif-else` statement

Structure: `if` statement

```
if order > 25:
    print("Free shipping!")
```

If this condition is `True`

Execute this code!

Example: `if` statement

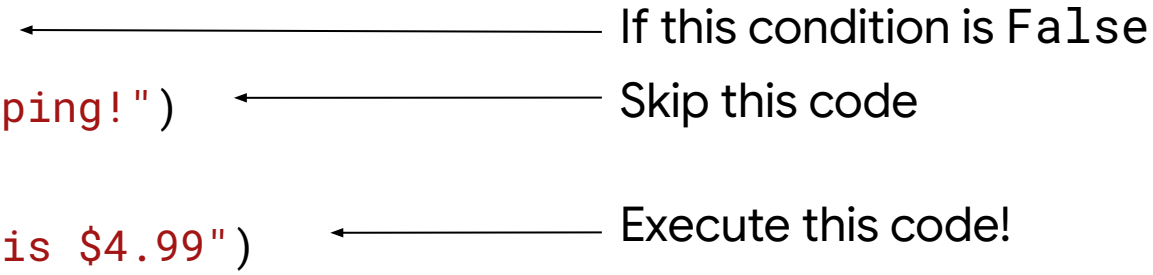
```
order = 34.50

if order > 25:
    print("Free shipping!")
```

```
Free shipping!
```

Structure: `if-else` statement

```
if order > 25:
    print("Free shipping!")
else:
    print("Shipping is $4.99")
```



Example: `if-else` statement

```
order = 22.50

if order > 25:
    print("Free shipping!")
else:
    print("Shipping is $4.99")

Shipping is $4.99
```

Examine the code

The user enters 4 for the input.

What is the output of this code block?

```
place = int(input("How did you place in the race?"))

if place == 1:
    print("You earned 10 points.")

else:
    print("You earned 0 points. Better luck next time!")
```

Examine the code

The user enters 4 for the input.

What is the output of this code block?

```
place = int(input("How did you place in the race?"))

if place == 1:
    print("You earned 10 points.")

else:
    print("You earned 0 points. Better luck next time!")
```

```
You earned 0 points. Better luck next time!
```

Structure: `if-elif-else` statement

```
if order > 25:
    print("Free shipping!")
elif order >= 20 and order <= 25:
 print("Shipping is $1.99")
else:
    print("Shipping is $4.99")
```

If this condition is `False`

Skip this code

If this condition is `True`

Execute this code!

Example: `if-elif-else` statement

```
order = 22.50

if order > 25:
    print("Free shipping!")
elif order >= 20 and order <= 25:
    print("Shipping is $1.99")
    print("Spend $25.00 for free shipping!")
else:
    print("Shipping is $4.99")

Shipping is $1.99
Spend $25.00 for free shipping!
```

Structure: nested `if` statement

```
if has_bogo_coupon:
    if order1 > order2:
        print(f"Using coupon! Total due: ${order1}")
    else:
        print(f"Using coupon! Total due: ${order2}")
else:
    print(f"No coupons. Total due: ${order1 + order2}")
```

If outer condition is `True`, check inner condition

If inner condition is `True`

Execute this code!

Example: Nested `if` statement

```
order1 = 25.75
order2 = 20.50

# Determines if customer has buy-one-get-one coupon
has_bogo_coupon = True

if has_bogo_coupon:
    if order1 > order2:
        print(f"Using coupon! Total due: ${order1}")
    else:
        print(f"Using coupon! Total due: ${order2}")
else:
    print(f"No coupons. Total due: ${order1 + order2}")

Using coupon! Total due: $25.75
```

Check your understanding

The user enters 57 for the input.
What is the output of this code block?

```
temp = int(input("What temperature is it outside? "))

if temp > 70:
   print("Wear a t-shirt.")

elif temp >= 58 and temp <= 69:
   print("Wear a light sweater.")

elif temp >= 45 and temp <= 57:
   print("Wear a sweater and a jacket.")

else:
   print("Wear a sweater and a coat.")
```

Check your understanding

The user enters 57 for the input.

What is the output of this code block?

```
temp = int(input("What temperature is it outside? "))

if temp > 70:
   print("Wear a t-shirt.")

elif temp >= 58 and temp <= 69:
   print("Wear a light sweater.")

elif temp >= 45 and temp <= 57:
   print("Wear a sweater and a jacket.")

else:
   print("Wear a sweater and a coat.")
```

```
Wear a sweater and a jacket.
```

IT'S CODING TIME

- Deciding the vibe for today's weather based on temperature
- Calculating if we reach our savings goals
- Determine if the a year, is a leap year

Questions?

Post to our Discord server

or

email:

upr-ciic3015-staff@upr.edu

karelys.lopez1@upr.edu

Give us anonymous feedback!

https://bit.ly/CIIC3015_F25_Feedback

CODE PATH *ORG